Future Generation Computer Systems 109 (2020) 317–330

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# Adaptive learning of aggregate analytics under dynamic workloads

Fotis Savva [a,*], Christos Anagnostopoulos [a], Peter Triantafillou [b]

[a] University of Glasgow G12 8QQ, UK
[b] University of Warwick CV4 7AL, UK

ARTICLE INFO

*Article history:*
Received 6 November 2019
Received in revised form 26 February 2020
Accepted 31 March 2020
Available online 7 April 2020

*Keywords:*
Life-long learning
Approximate query processing
Machine learning
Concept drift detection
Change-point detection

ABSTRACT

Large organizations have seamlessly incorporated data-driven decision making in their operations. However, as data volumes increase, expensive big data infrastructures are called to rescue. In this setting, analytics tasks become very costly in terms of query response time, resource consumption, and money in cloud deployments, especially when base data are stored across geographically distributed data centers. Therefore, we introduce an adaptive, reciprocity-based Machine Learning mechanism which is light-weight, stored client-side, can estimate the answers of a variety of aggregate queries and can avoid the big data back-end. The estimations are performed in milliseconds are inexpensive and accurate as the mechanism learns from past analytical-query patterns. However, as analytic queries are ad hoc and analysts' interests change over time we develop solutions that can swiftly and accurately detect such changes and adapt to new query patterns. The capabilities of our approach are demonstrated using extensive evaluation with real and synthetic datasets.



## 1. Introduction

With the rapid explosion of data volumes and the adoption of data-driven decision making, organizations have been struggling to process data efficiently. Because of that a surge of companies is turning to popular cloud providers that have created large-scale systems capable of storing and processing large data quantities. However, the problem still remains in that multiple analytics queries are issued by multiple analysts (Fig. 1) which often overburden data clusters and are costly. Data analysts should be able to extract information without significant delays so as not to violate the *interactivity constraint* set around 500 ms [1]. Anything over that limit can negatively affect analysts' experience and productivity. This constraint is particularly important in the context of *exploratory analysis* [2]. Such analyses are an invariable step in the process of understanding data and creating solutions to support business decisions. Furthermore, aggregate analytics are becoming increasingly geo-distributed, which are time consuming and nearly impossible when data have to remain federated without the possibility of transferring them to central locations [3]. Same applies to sensitive data that can only be accessed via aggregate queries with no data samples allowed.

**Vision:** Depicted at Fig. 1 is our vision for an aggregate analytics learning & prediction system that is light-weight, stored on an Analyst's Device (AD) and adaptive to dynamic query workloads. This allows the exploratory process to be executed locally at ADs providing predictions to aggregate queries thus not overburdening the Cloud/Central System (CS). Prediction-based aggregate analytics is expected to save computational and communication resources, which could be devoted to cases where accurate answers to aggregate queries are demanded. From the CS's perspective, our system acts as a *pseudo*-caching mechanism to reduce communication overhead and computational load when it is necessary, thus, allowing for other tasks/processes to run.

Our system offers a learning-based, prediction-driven way of performing aggregate analytics in ADs accessing *no data*. It neither requires data transmission from CS to ADs nor from ADs to CS. What makes such a system possible is the exploitation of previously executed queries and their answers sitting in log files. We adopt Machine Learning (ML) regression models that learn to associate past executed queries with their answers, and can in turn locally predict the answers of *new* queries. Subsequent aggregate queries are answered in milliseconds, thus, fulfilling the interactivity constraint.

Furthermore, our framework can directly adapt to analysts' (dynamic) query workloads by monitoring the analysts' query patterns and adjusting their parameters. Shown at Fig. 1 are the ML models $f$ and mechanisms developed for detecting and adapting to changes in query patterns. Both of them are discussed in Sections 3 and 4.

**Challenges & Contribution:** A large number of analysts exist within an organization with diverse analytics interests thus their query patterns are expected to differ, accessing different parts

* Corresponding author.
*E-mail addresses:* f.savva.1@research.gla.ac.uk (F. Savva), christos.anagnostopoulos@glasgow.ac.uk (C. Anagnostopoulos), p.triantafillou@warwick.ac.uk (P. Triantafillou).

https://doi.org/10.1016/j.future.2020.03.063

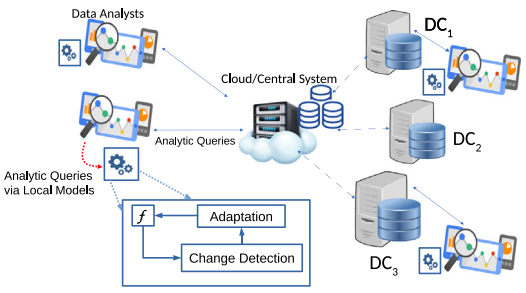



**Fig. 1.** Aggregate analytics eco-system with analysts' devices, data centers, and local adaptive ML models.

of the whole data-space. We are challenged to support model training over vastly different patterns, which are to be drastically changing or expanding in dynamic environments. Moreover the models have to be up-to-date w.r.t. pattern changes, which require early query pattern change detection and efficient adaptation. Given these challenges, our contributions are:

1. A novel query-driven mechanism and query representation that associates queries with their respective answers and can be used by ML models.
2. A local change detection mechanism for detecting changes in query patterns based on our prediction-error approximation.
3. A reciprocity-based adaptation mechanism in light of novel query patterns, which efficiently engages the CS to validate the prediction-to-adaptation states transition and guarantees system convergence.
4. Comprehensive assessment of the system performance and sensitivity analysis using real and synthetic data and query workloads.

## 2. Fundamentals of query-driven learning

The fundamentals of query-driven mechanism for analytics are: (1) transforming analytic queries in a real-valued vectorial space, (2) quantization of vectorial space, extracting query patterns and (3) training of local regression models for predicting query answers using past issued queries. Principally, we learn to associate the result of a query using the derived query patterns and linking these patterns with local regression models. Given an unseen query, we project it to the closest query pattern we have learned and then predict its corresponding result without executing the query over the data in a DC/CS. Readers familiar with the Query-Driven Learning methodology should feel free to skip this section.

**Definition 1.** A dataset $\mathcal{B} = \{\mathbf{a}\}_{i=1}^{n}$ is a set of $n$ row data vectors $\mathbf{a} = [a_1, \ldots, a_d] \in \mathbb{R}^d$ with real attributes $a_i \in \mathbb{R}$.

Analytics queries are issued over a $d$-dimensional data space and bear two key characteristics: First, they define a subspace of interest, using various predicates on attribute (dimension) values. Second, they perform aggregate functions over said data subspaces (to derive key statistics over the subspace of interest). We adopt a general vectorial representation for modeling a query over any type of data storage/processing system.

Predicates over attributes define a data subspace over $\mathcal{B}$ formed by a sequence of logical conjunctions using (in)equality constraints ($\leq, \geq, =$). A *range-predicate* restricts an attribute $a_i$ to be within range $[l_i, u_i]$: $a_i \geq l_i \wedge a_i \leq u_i$. We model a range query over $\mathcal{B}$ through conjunctions of predicates, i.e., $\bigwedge_{i=1}^{d}(l_i \leq a_i \leq u_i)$ represented as a vector in $\mathbb{R}^{2d}$.

**Definition 2.** A (range) row query vector is defined as $\mathbf{q} = [l_1, u_1, \ldots, l_d, u_d] \in \mathbb{R}^{2d}$ corresponding to the range query $\bigwedge_{i=1}^{d}(l_i \leq a_i \leq u_i)$. The distance between two queries $\mathbf{q}$ and $\mathbf{q}'$ is defined as the $L_2^2$ norm (Euclidean distance): $\|\mathbf{q}-\mathbf{q}'\|_2^2 = \sum_{i=1}^{d}(l_i - l_i')^2 + (u_i - u_i')^2$.

This representation is flexible enough to accommodate a wide variety of queries. As the dimensionality of the query vector is proportional to the data vector, queries with predicates bounding the values of different attributes can be used by the same ML algorithm. This means that only a number of $(l_i, u_i)$ values are set w.r.t the number of predicates for a given query. In addition, we make no assumptions as to the back-end system as what is being parsed are the filters in a query. This allows the mechanism to be deployed in parallel to multiple analytic systems.

In query-driven learning, we learn to associate a query with its corresponding aggregate result (a scalar $y \in \mathbb{R}$). This is achieved using a *training* set of query–result pairs $\mathcal{T} = \{(\mathbf{q}_i, y_i)\}_{i=1}^{N}$ obtained after executing $N$ queries over dataset $\mathcal{B}$. The goal is to develop an ML model based on $\mathcal{T}$ to minimize the expected prediction error between actual $y$ and predicted $\hat{y}$, $\mathbb{E}[(\hat{y}-y)^2]$ and predict the result of any unseen query *without* executing it over $\mathcal{B}$.

### 2.1. Query space clustering

Recent research analyzing analytics workloads from various domains has shown that queries within analytics workloads share patterns and their results are *similar* having various degrees of overlap [4]. Based on this evidence, we mine query logs (the *training* set) and discover clusters of queries (in the vectorial $d$-dimensional space), having similar predicate parameters. This partitioning is fundamental to get accurate ML models for predictive analytics, as we then associate different ML predictive models with different clusters. In this way, learning different data subsets is proven to be more efficient in terms of *explainability/model-fitting* and *predictability* than having one global ML model learn everything and is also [5] known as *ensemble learning* [6]. To put this in context, consider a discrete time domain $t \in \mathbb{T} = \{1, 2, 3, \ldots\}$, where at each time instance $t$ an analyst issues a query $\mathbf{q}_t$. The query is executed and an answer $y_t$ is obtained, forming the pair $(\mathbf{q}_t, y_t)$. The issued queries are stored in a growing set $\mathcal{C}_t = \{(\mathbf{q}_1, y_1), \ldots, (\mathbf{q}_t, y_t)\} = \mathcal{C}_{t-1} \cup \{(\mathbf{q}_t, y_t)\}$. Given this set, we incrementally extract knowledge from the query vectors and then train *local* ML models that predict the associated outputs given new, unseen queries. This is achieved by on-line partitioning the vectors $\{\mathbf{q}_1, \ldots, \mathbf{q}_t\} \in \mathcal{C}_t$ into disjoint clusters that represent the query patterns of the analysts (fundamentally, within each cluster the queries are much more *similar* than the queries in other clusters). The distance between queries quantifies how close the predicate parameters are in the vectorial space. Close queries $\mathbf{q}$ and $\mathbf{q}'$ are grouped together into $K$[1] clusters w.r.t. $\|\mathbf{q} - \mathbf{q}'\|_2^2$. The objective is to minimize the expected quantization error $\mathbb{E}[\min_{k=1,\ldots,K} \|\mathbf{q} - \mathbf{w}_k\|_2^2]$ of all queries to their closest cluster *representative* $\mathbf{w}_k$, which reflects the analysts query patterns and best represents each cluster. The $K$ query representatives $\mathcal{W} = \{\mathbf{w}_1, \ldots, \mathbf{w}_K\}$ optimally quantize $\mathcal{C}_t$ minimizing the expected quantization error while each query $\mathbf{q}$ is projected onto its closest representative $\mathbf{w}^* = \arg\min_{\mathbf{w}\in\mathcal{W}} \|\mathbf{q} - \mathbf{w}\|_2^2$. Based on the partitioning of $\mathcal{C}_t$, we produce $K$ query-disjoint sub-sets such that $\mathcal{C}_k \cap \mathcal{C}_l \equiv \emptyset$ for $k \neq l$ and $\mathcal{C}_k = \{(\mathbf{q}, y) \in \mathcal{C}_t | \mathbf{w}_k = \arg\min_{\mathbf{w}\in\mathcal{W}} \|\mathbf{q}-\mathbf{w}\|_2\}$. A *local* ML model is then trained over each subset using the pairs in $\mathcal{C}_k$, $k \in [K]$.

[1] The number of clusters $K$ is automatically identified by the clustering algorithm used [7].

### 2.2. Query–answer predictive model

Each aggregate result $y$ from the pair $(\mathbf{q}, y) \in \mathcal{C}_t$ represents the *exact* answer produced by the CS. Essentially, $y$ is produced by an unknown function $g$ which we wish to learn. Such function produces query-answers w.r.t an unknown distribution $p(y|\mathbf{q})$. Our aim is to approximate the *true* functions $g$ for aggregate functions (descriptive statistics) e.g., count, average, max, sum etc. Regression algorithms are trained using query–answer pairs from $\mathcal{C}_t$ to minimize the expected prediction error between actual $y = g(\mathbf{q})$, from the true function $g$, and predicted $\hat{y}$ from an approximated function $\hat{g}$, i.e., $\mathbb{E}[(g(\mathbf{q}) - \hat{g}(\mathbf{q}))^2]$. After having partitioned the query space into clusters $\mathcal{C}_1, \ldots, \mathcal{C}_K$, we therein train $K$ local ML models, $\mathcal{M} = \{\hat{g}_1, \ldots, \hat{g}_K\}$ that associate queries $\mathbf{q}$ belonging to cluster $\mathcal{C}_k$ with their outputs $y$. Each ML model $\hat{g}_k$ is trained from query–response pairs $(\mathbf{q}, y) \in \mathcal{C}_t$ from those queries $\mathbf{q}$ which belong to $\mathcal{C}_k$ such that $\mathbf{w}_k$ is the closest representative to those queries. The originally trained ML models in DC/CS are then sent to ADs to be used for predicting answers. Given a query $\mathbf{q}$ only the most representative model $\hat{g}_k$ is used for prediction, corresponding to the closest $\mathbf{w}_k$:

$$\hat{y} = \sum_{k=1}^{K} \mathcal{I}_k \hat{g}_k(\mathbf{q}) \tag{1}$$

where $\mathcal{I}_k = 1$ if $\mathbf{w}_k = \arg\min_{\mathbf{w} \in \mathcal{W}} \|\mathbf{q} - \mathbf{w}\|_2^2$; 0 otherwise.

## 3. Query pattern change detection

Suppose that all trained ML models $\{\hat{g}_k\}_{k=1}^K$ are delivered to the analysts from CS, indicating that the mechanism enters its *prediction mode*. That is, for each incoming query, it predicts the answer and delivers it back to the analysts *without* query execution. If we assumed a stationary query pattern distribution, in which queries and analysts' interests do not change, then this would suffice. However, this is not realistic as it is highly likely that *analysts interests* change over time (e.g., during exploratory analytics tasks, which are considered as ad hoc processes [2]). So, dynamic workloads may render the $\{\hat{g}_k\}_{k=1}^K$ models obsolete, as they were trained using *past* query patterns following distributions which may now be different. Accommodating such dynamics is becoming increasingly important as ML is widely adopted in software in production [8]. Specifically, when referring to analysts' interests, we refer to analysts who are tasked with informing different business decision processes. If those tasks change, the data subspaces to be analyzed become different, which results in changed query patterns. If models cannot be adaptive, expected prediction errors can become arbitrarily high. When $p(y|\mathbf{q})$ changes to $p'(y|\mathbf{q})$, it is highly likely that any previous approximation would produce high-error answers, unless $p(y|\mathbf{q}) \approx p'(y|\mathbf{q})$. We capture such dynamics as *concept drift* [9,10] – many methods have been developed for adjusting when this arises [10,11].

We introduce a Change Detection Mechanism (CDM) and an Adaptation Mechanism (ADM) (shown at Fig. 1) addressing this concern raising a number of challenges: (1) How to detect a query pattern change; we need to enable triggers that alert the mechanism being in prediction mode in case of a *concept drift*; (2) What kind of action should we take in case that happens, i.e., what strategy to follow for updating the ML models; (3) How should we notify users, analysts, and applications about such change(s) or even who to notify; shall we transmit an update to all users or just the affected ones? We explore these challenges and the describe the decisions we take in tackling them in the remainder.

### 3.1. Change detection mechanism

So far, we have trained $K$ different local ML models to predict answers involving only the $k$th model that best represents a new incoming query through the representative $\mathbf{w}_k$. This requires to individually monitor whether the query representatives, used for prediction via their respective models, are *still* representatives in long-term predictions or whether the analysts' query patterns have changed. In this case, we need to introduce a CDM that triggers when the original query representative has significantly diverged from the estimated one.

Our approach can be best understood by first assuming that the CDM maintains an on-line average of the prediction error $(y - \hat{y})^2$ such that : $u_k \approx \mathbb{E}[(y - \hat{y})^2|\mathbf{q}]$. This is done for each query representative $\mathbf{w}_k$ across different users. Should the expected error $u_k$ escalate significantly, then this may signal that a query pattern has shifted around the 'region' represented by the representative $\mathbf{w}_k$. But, recall that during the prediction mode, the actual $y$ is unknown since our goal is to predict accurate answers but without executing the query itself. Hence, we develop an approximation mechanism for change detection, not requiring query executions over CS/DC.

Once we have trained the individual ML models $\mathcal{M}$ and calculated their expected prediction accuracy (using an independent test sample drawn from the original set of queries) we obtain the Expected Prediction Error (EPE), which will be constant across all possible queries associated with a particular query representative defined as: $EPE = \mathbb{E}\left[\left(g(\mathbf{q}) - \sum_{\kappa=1}^{K} \mathcal{I}_\kappa \hat{g}_\kappa(\mathbf{q})\right)^2\right]$. Using the EPE, we wish to find a fine-grained estimate of the true prediction error rather just assuming this is constant for each and every unseen query.

To do this, we have analyzed the error behavior under changing query patterns. Our findings reveal an interesting fact: The Euclidean distance $d(\mathbf{q}, \mathbf{w}_k) = \|\mathbf{q} - \mathbf{w}_k\|_2^2$ of a random query $\mathbf{q}$ from its closest query representative $\mathbf{w}_k$ is strongly correlated with the associated prediction error $(y - \hat{y})^2$.[2] Considering the correlation between $d(\mathbf{q}, \mathbf{w}_k)$ and the local $u_k$, we define a distance-based prediction error $\tilde{u}_k$ of a query $\mathbf{q}$ as:

$$\tilde{u}_i = \ln(1 + d(\mathbf{q}, \mathbf{w}_k) - \min_{\mathbf{q}_\ell \in \mathcal{C}_k} d(\mathbf{w}_k, \mathbf{q}_\ell)) \cdot u_k, \tag{2}$$

where the natural-log operator acts as a penalizing/discount factor for queries given their distance from the closest representative $\mathbf{w}_k$. The second term within the natural-log operator, $\min_{\mathbf{q}_\ell \in \mathcal{C}_k} d(\mathbf{w}_k, \mathbf{q}_\ell)$ is the minimum distance between the query representative $\mathbf{w}_k$ and the associated queries $\mathbf{q} \in \mathcal{C}_k$. We subtract the minimum distance from $d(\mathbf{q}, \mathbf{w}_k)$ so that the scale of the numbers will not affect the computation of the error.

We base our novel CDM in (2) using the series of error approximations $\{\tilde{u}_t\}$ for monitoring concept drifts in query patterns during prediction mode without executing the queries.

Consider the incoming unseen (random) queries $(\mathbf{q}_0, \mathbf{q}_1, \ldots, \mathbf{q}_t)$ arriving in a sequence $t \in \mathbb{T}$ during prediction mode. They are answered by specific local ML models $(\hat{g}_0, \hat{g}_1, \ldots, \hat{g}_k)$, generating a series of distance-based error estimations $\{\tilde{u}_t\}$, $t \in \mathbb{T}$. The CDM monitors this series and, based on a specific threshold, signals the existence of *concept drift*, i.e., checks whether the probability distribution of the queries has changed. Based on the series of error estimations, we learn two query distributions: (1) the *expected* query distribution, which is represented by the query representatives and (2) the *novel* query distribution, which cannot be represented by the current query representatives. The expected distribution $p_0(\tilde{u})$ is estimated given a training period

[2] A 0.3 Pearson's Correlations was obtained on a real dataset.

from $\tilde{u}_k$ values corresponding to queries with closest representative $\mathbf{w}_k$. The novel distribution $p_1(\tilde{u})$ is estimated from $\tilde{u}_m$ values corresponding to error values derived from the *rival* representatives $\mathbf{w}_m$ of queries with closest $\mathbf{w}_m$ and $k \neq m$. Based on this, we estimate the distribution of the error values generated from representatives which were not the closest to the queries, thus, approximating novel error values. Both distributions were approximated by fitting the $p(\tilde{u}) \sim \Gamma(e_1, e_2)$ distribution with scale $e_1$ and shape $e_2$.

Given a $\tilde{u}_t$ value, we calculate the likelihood ratio $s_t = \log \frac{p_1(\tilde{u}_t)}{p_0(\tilde{u}_t)}$ and the cumulative sum of $s_t$ up to time $t$, $U_t = \sum_{\tau=0}^{t} s_\tau$. Based on the sequential ratio monitoring for a progressive concept drift in distribution [12] from $p_0$ to $p_1$, a decision function is introduced for signaling a potential concept drift expressed as:

$$G_t = U_t - \min_{0 \leq \tau \leq t} U_{\tau-1}. \tag{3}$$

The decision function in (3) indicates the current cumulative sum of ratios minus its current minimum value. This denotes that the *change time* estimate $t^*$ is the time following the current minimum of the cumulative sum, i.e., $t^* = \arg\min_{0 \leq \tau \leq t} U_\tau$. Therefore, given that $U_t = U_{t-1} + s_t$, the decision function in (3) this is re-written in a recursive form: $G_t = \{G_{t-1} + s_t\}^+$ with $\{z\}^+ = \max(z, 0)$ setting, by convention, $U_{-1} = 0$ and $G_{-1} = 0$. Hence, a concept drift of query patterns projected over the query representatives space is detected at time $t_D$: $t_D = \min\{t \geq 0 : G_t > h\}$. The parameter $h$ is usually set $3\sigma \leq h \leq 5\sigma$ with $\sigma$ the standard deviation of $\tilde{u}$. The process is shown at Fig. 2, the cumulative sum of ratios exceeds the threshold $h$ as soon as queries are issued from an unknown distribution as the error estimates become steadily larger and are not just random fluctuations in errors. It is worth noting that the change in query distribution is based on fusing the distance between the queries and their closest representatives scaled with the expected prediction error. We refer to this as an indication of degradation in the performance of the model. Given that a change has been detected, the CDM signals the ADM which transits from *prediction mode* to *buffering mode* as shown at Fig. 3. As soon as a change is detected the CDM signals the ADM component, that new query patterns have been detected. In turn, the ADM signals the *Prediction Component* (containing the $\mathcal{M}$ and $\mathcal{W}$) to be put in `BUFFERING` mode since the prediction component can no longer provide reliable answers for *all* queries. However, the AD can still leverage the complete system to ask queries following the already known distributions with *only* queries following the new shifted distribution being executed at the CS. By entering `BUFFERING` mode our ADM starts to adjust for the new query patterns under the AD until *converging*. At that point it signals the *Prediction Component* to switch back to `PREDICTION` mode, resuming normal operation.

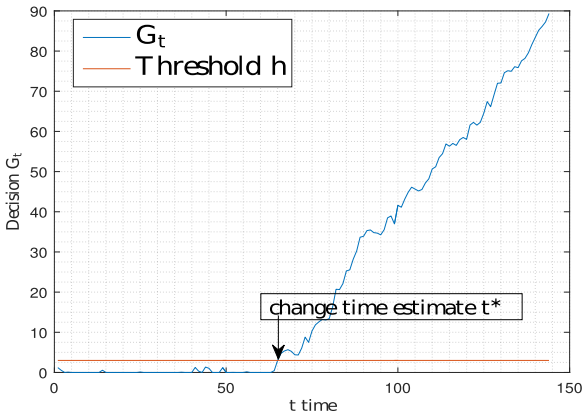



**Fig. 2.** Change detection based on the likelihood ratio of the distance-based error triggered when query patterns are shifted.

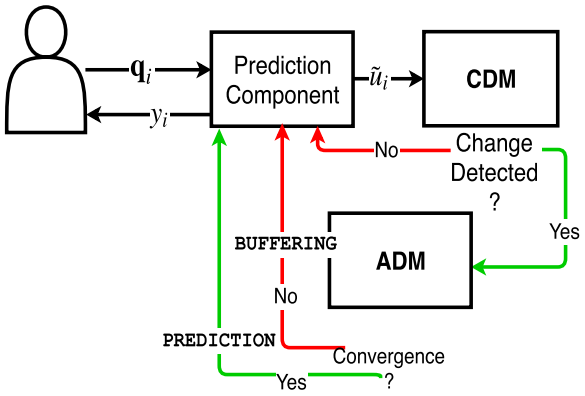



**Fig. 3.** Overview of the operation of CDM and ADM which control when estimations can be reliably given to the analyst.

## 4. Model adaptation

In this section, we explain the fundamentals of the ADM along with unintended results we can exploit. Once a local model $\hat{g}_k$ transits to buffering mode, it is deemed unreliable to accurately predict the answers to incoming queries. Therefore, during this phase, queries should be executed and their actual answers returned to analysts while also being used for adapting the model and representative $(\hat{g}_k, \mathbf{w}_k)$. In the beginning of buffering mode, $\hat{g}_k$ and $\mathbf{w}_k$ are sent to CS, which will tune/adapt their parameters using the actual execution of queries. To reduce the expected number of queries executed in CS during buffering mode, we introduce a *query execution selectivity* mechanism based on the current estimated error in (2). Specifically, there would still be some queries issued by the AD that could be locally answered by current models cached in AD during that phase. Therefore, the AD still monitors incoming queries and *discriminates* between two types: (1) the ones that can be locally answered by models in $\mathcal{M}$ and (2) the ones that cannot be answered, since these queries are not well represented by the cached query representatives. The latter queries are then forwarded to CS for execution. The selectivity mechanism relies on the following rule: an incoming query $\mathbf{q}_t$ at an AD in buffering mode is locally answered if the distance from the query representative of the new query patterns, notated by $\mathbf{w}_{K+1}$, is not the closest representative i.e., $\mathbf{w}_{k^*} = \arg\min_{\mathbf{w} \in \mathcal{W} \cup \{\mathbf{w}_{K+1}\}} \|\mathbf{q}_t - \mathbf{w}\|_2$ and $k^* \neq K+1$. If the query is closest to the non-yet converged novel representative, then it is forwarded to CS for execution. However, since the novel representative $\mathbf{w}_{K+1}$ is not converged, we also consider the distance from its rival (second closest) converged representative as a backup. The rival representative can provide assistance and answer the query locally instead of forwarding it to the CS if it is close enough to include $\mathbf{q}_t$ in the range around its variance $\sigma^2$.[3] An example is shown at Fig. 4, queries $\mathbf{q}_t$ and $\mathbf{q}_{t+1}$ both have $\mathbf{w}_{k+1}$ as the closest representative. However, only $\mathbf{q}_{t+1}$ will be forwarded as $\mathbf{q}_t$ is within the radius of $\mathbf{w}_3$. The forwarding selectivity mechanism is also evident in Algorithm 1 (lines 5–10). Based on the centroid theorem of convergence in vector quantization, i.e., the converged $\mathbf{w}_k$ is the expected query (centroid) of those queries having $\mathbf{w}_k$ as their closest representative, we

[3] This is associated with the vigilance parameter in Adaptive Resonance Theory dealing with the bias-plasticity dilemma.

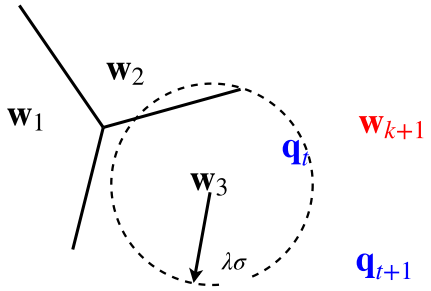



**Fig. 4.** Demonstration of the *forwarding rule* by the ADM. Both queries (in blue) have $\mathbf{w}_{k+1}$ (the non-converged) as the closest representative. Only one of them is send to the CS. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

exploit the variance $\sigma^2 = \frac{1}{|\mathbb{Q}_k|}\sum_{\mathbf{q}\in\mathbb{Q}_k}\|\mathbf{q}-\mathbf{w}_k\|_2^2$ for activating the forwarding rule. The rule is based on the rival centroid and is fired if $\|\mathbf{q}_t - \mathbf{w}_k\|_2 > \lambda\sigma$ for any scalar $\lambda > 0$ given that the query is closest to the non-converged $\mathbf{w}_{K+1}$. The probability of forwarding incoming queries from the AD to CS for execution, given that they cannot be reliably answered by the local model $\hat{g}_k$ is upper bounded as provided in Theorem 1. The query is executed if inevitably the rival representative cannot be used for prediction since $\|\mathbf{q}_t - \mathbf{w}_{K+1}\|_2 > \lambda\sigma$. The value of $\lambda$ is adopted from the scaling factor of $h$, i.e., $3 \leq \lambda \leq 5$.

**Theorem 1.** *Given a random query* $\mathbf{q}$ *whose distance from its rival (second closest) representative* $\mathbf{w}_k$ *is greater than* $\lambda\sigma$*, the upper bound of the forwarding probability for query execution is* $O(\frac{1}{\lambda^2})$.

**Proof.** Let the query $\mathbf{q}$ being projected to its closest representative $\mathbf{w}_{K+1}$, which is not yet converged and let its second closest be the converged $\mathbf{w}_k$. The representative $\mathbf{w}_k$ corresponds to the mean vector of those queries belonging in the cluster $\mathcal{C}_k$. In order to the query $\mathbf{q}$ to be forwarded to the CS for execution it means that the $\mathbf{w}_k$ should not be the mean vector for the incoming query $\mathbf{q}$. This is indicated if the distance $\|\mathbf{q}-\mathbf{w}_k\|_2$ is greater than a proportion of the norm of the variance $\sigma$ of the cluster $\mathcal{C}_k$ by a factor $\lambda > 1$. Hence, the query is sent from the AD to the CS if at least this distance is greater than $\lambda\sigma$, which is stochastically bounded by the factor $1/\lambda^2$ based on Chebyshev's inequality $P(\|\mathbf{q}-\mathbf{w}_k\|_2) \geq \lambda\sigma \leq \frac{1}{\lambda^2}$. □

### 4.1. Taking advantage of affiliates

In the ADM, we take advantage of the tuning process taking place in the CS. We exploit queries coming from the original AD which triggered the CDM and other queries coming from different ADs also in buffering mode due to some other *independent* triggered CDMs. We call these potential ADs, *affiliates* belonging to set $\mathcal{A}$, since their executed queries and actual answers are used for tuning the stale models. Let $n$ ADs be connected to CS and assume that each AD system $j$ referring to model $\mathcal{M}_j$ enters its buffering mode independently of the others with entry probability $\beta_j = P(G_j > h) \in (0, 1)$. Then, the probability of an AD (being in buffering mode) to *meet* at least one affiliate $j$ in CS is $(1 - \prod_{j=1}^{n-1}(1-\beta_j))$. The expected number of affiliates is then approximated by $\mathbb{E}[|\mathcal{A}|] \approx \beta(n-1)$ under the assumption that the entry probabilities are almost the same $\beta_j = \beta, \forall j$. This expectation will be used for studying the knowledge expansion in terms of novel query patterns being delivered to an AD via our reciprocity adaptation mechanism.

**Algorithm 1:** Reciprocity-based Model Adaptation in CS

```
1  CS receives (copy) of M and W from AD;
2  Set buffer Q = ∅, affiliate buffer Q_A = ∅;
3  while MODE = BUFFERING do
4      Prediction Component receives query q_t from AD;
5      w_{k*} = arg min_{w∈W∪{w_{K+1}}} ||q_t − w||_2 /*closest*/;
6      w_k = arg min_{w∈W∪{w_{K+1}}−{w_{k*}}} ||q_t − w||_2, /*rival*/;
7      if ||q_t − w_k||_2 > λσ and w_{K+1} = w_{k*} then
8          send query q_t to CS for execution;
9          Q = Q ∪ {(q_t, y_t)} /*actual query–answer pair*/ ;
10         adapt prototype w_{K+1} ;
11     else
12         ŷ = ĝ(q_t) /*prediction*/;
13     end
14     for affiliate j ∈ A do
15         receive affiliate query q_j;
16         if ||q_j − w_k||_2 ≤ λσ then
17             Q_A = Q_A ∪ {(q_j, y_j)} /*affiliate pair*/ ;
18         end
19     end
20     update learning rate γ;
21     if convergence w.r.t. c then
22         train new model ĝ_{K+1} using Q;
23     end
24     M = M ∪ {ĝ_{K+1}, ĝ_j}, W = W ∪ {w_{K+1}, w_j}, ∀j ∈ A;
25     M and W are sent to AD from CS;
26     set MODE = PREDICTION
27 end
```

### 4.2. Model adaptation & reciprocity

In the CS, when a query is selectively forwarded from AD, the process of model adaptation has as follows: for adapting to new query patterns, we rely on the principle of *explicit partitioning* [9,10], as a natural extension of our strategy using an ensemble of local ML models. To adjust to new query patterns, we train a new model $\hat{g}_{K+1}$ using executed queries and their answers in CS. This is the optimal strategy for expanding the current $\mathcal{M}$ as other methods might lead to *catastrophic forgetting* [11]. Indicatively, such methods adopt strategies to adapt the current model by adjusting to new patterns whilst forgetting the old ones. In our context, this is not applicable since analysts have the flexibility to issue queries either conforming to the *old* patterns or to the new ones, depending on the analytics process.

The adaptation process is performed with parameters: the $K$ query prototypes $\mathcal{W}$ and their associated ML models $\mathcal{M}$ as shown in Algorithm 1. Recall that the analyst's device has cached models $\mathcal{M}$ and the DC/CS adapts the received parameters by learning the new underlying query patterns and based on these trains the new ML model. Let the queries series $\{\mathbf{q}_1, \mathbf{q}_2, \ldots\}$ coming from the AD to CS based on selective forwarding. This means that most likely a query $\mathbf{q}_t$ conforms to new query patterns thus sent to CS for execution. Once query $\mathbf{q}_1$ is executed and its actual answer $y_1$ is obtained, it is then considered as a new (initial) representative $\mathbf{w}_{K+1}$ for $\mathcal{M}_{K+1}$. The pairs $(\mathbf{q}_t, y_t)$ are then used to incrementally update $\mathbf{w}_{K+1}$ and then buffered in $\mathcal{Q}$, which will be the training set for $\hat{g}_{K+1}$ (lines 4–13). The adaptation of $\mathbf{w}_{K+1}$ to follow the new query pattern is achieved by Stochastic Gradient Descent (SGD) [13], which is widely used in statistical learning for training in an on-line manner considering one training example (query–answer) at a time. We focus on the convergence of the query distribution by moving the new query representative towards the estimated median of the queries in $\mathcal{Q}$ and not the corresponding centroid. This is introduced so that the new representative

converges to a robust statistic, free of outliers and more reliable than the centroid (mean vector). The convergence to the median denotes with high reliability *convergence to the distribution*, which is what we desire for model convergence. In this case, we provide the adaptation rule of the new query representative to converge to the median of the forwarded queries, as provided in Theorem 2.

**Theorem 2.** *The novel representative* $\mathbf{w}_{K+1}$ *converges to the median vector of the queries executed in the DC w.r.t. update rule* $\Delta\mathbf{w}_{K+1} \propto \gamma sgn(\mathbf{q} - \mathbf{w}_{K+1})$, $\gamma \in (0, 1)$; $sgn(\cdot)$ *is the signum function.*

**Proof.** Each dimension $i$ of the median vector $\mathbf{m}$ of the queries $\mathbf{q}$ in a sub-space satisfies: $P(q_i \geq m_i) = P(q_i \leq m_i) = \frac{1}{2}$. Suppose that the new representative $\mathbf{w}_{K+1}$ has reached equilibrium, i.e., $\Delta\mathbf{w}_{K+1} = \mathbf{0}$ holds with probability 1. By taking the expectations of both sides of the update rule $\mathbb{E}[\Delta\mathbf{w}_{K+1}] = \alpha\mathbb{E}[sgn(\mathbf{q} - \mathbf{w}_{K+1})] = \mathbf{0}$ and focusing on each dimension $i$, we obtain that: $\int sgn(q_i - w_{K+1,i})p(q_i)dq_i = P(q_i \geq w_{K+1,i}) \int p(q_i)dq_i - P(q_i < w_{K+1,i}) \int p(q_i)dq_i = 2P(q_i \geq w_{K+1,i}) - 1$. Since $\mathbb{E}[\Delta w_{K+1,i}] = 0$ is constant, then $P(q_i \geq w_{K+1,i}) = \frac{1}{2}$, which denotes that $w_{K+1,i}$ converges to the median of $q_i$, $\forall i$. □

Given the $j$th incoming query $\mathbf{q}_j$ issued by analysts in the $j$th affiliate AD, the CS assesses the selectivity forwarding criterion: $\|\mathbf{q}_j - \mathbf{w}_k\|_2 \leq \lambda\sigma$. If it holds, these affiliate queries are exploited for expanding the query patterns (lines 14–18). That is, CS buffers the pairs $(\mathbf{q}_j, y_j)$ in affiliate set $\mathcal{Q}_\mathcal{A}$ which will be used later for training new ML models $\hat{g}_j$ enriching the predictability variety of $\mathcal{M}$. In this case, we obtain the affiliate new representative $\mathbf{w}_j$ generated by query patterns coming from the affiliate AD $j \in \mathcal{A}$. Similarly to the new $\mathbf{w}_{K+1}$, affiliate $\mathbf{w}_j$ is incrementally adapted through SGD with the aim to converge to the corresponding median of the affiliate queries in $\mathcal{Q}_\mathcal{A}$. For the new $\mathbf{w}_{K+1}$ and the possibly affiliate $\mathbf{w}_j$, the median convergence rule involves the learning rate:

$$\gamma = \frac{1}{1 + |\mathcal{Q}| + |\mathcal{Q}_\mathcal{A}|}, \tag{4}$$

which decreases as more queries are appended to $\mathcal{Q}$ and $\mathcal{Q}_\mathcal{A}$; the higher the number of affiliate query representatives, the faster the convergence to the median. This demonstrates the exploitation of affiliates to the adaptation of $\mathcal{M}$.

The convergence of the representatives is checked by the subsequent adjustments in positions that $\mathbf{w}_{K+1}$ makes. If that change is lower than a threshold $c$ then convergence has been achieved. After the convergence of the query representative and affiliates (if any), the CS trains the new models $g_{\hat{K+1}}$ and $\{\hat{g}_j\}$, using the $\mathcal{Q}$ and $\mathcal{Q}_\mathcal{A}$, respectively. The new ML models and new representatives are then delivered to AD (lines 20–26). Evidently, set $\mathcal{W}$ is now expanded with one more representative $\mathbf{w}_{K+1}$ and on average $(n-1)\beta$ affiliate representatives along with their regression models. The adapted and updated $\mathcal{M}$ is expected to have $K + 1 + (n-1)\beta$ query representatives and ML models at the end of the buffering phase.

### 4.3. Convergence to an offline mode

When the system transits from the buffering to prediction mode, the enhancement of $\mathcal{M}$ and $\mathcal{W}$ gradually decreases the probability to enter the buffering mode in the future, in light of learning the query patterns not only derived from the analysts interacting with the CS/DC, but also the patterns from other analysts in other ADs. This indicates that the gradually expanding sets reflect the analysts' way of exploring and analyzing data among data centers. Because of this expansion, the transition probability from prediction to buffering mode gradually decreases saving computational and communication resources at the network and CS. The expected ratio of new models in an AD transiting from the $m$th buffering mode with $K_m$ representatives to the $(m+1)$th prediction mode is: $1 + \frac{1+\beta(n-1)}{K_m}$, with $\mathbb{E}[\Delta K_m] = \mathbb{E}[K_{m+1} - K_m] = 1 + \beta(n-1)$. Such ratio increases to unity after certain prediction-buffering transitions denoting that *all* query sub-spaces are known:

$$\lim_{m\to\infty}(1 + \frac{1 + \beta(n-1)}{m}) = 1. \tag{5}$$

An AD model then learns *all* possible query sub-spaces via its analysts and affiliate models with rate $O(\frac{1}{m})$. The entry probability to buffering mode decreases with the same rate, thus, reducing the CS execution overhead and communication load by transiting the AD to 'offline' mode. This is the advantage of the query-driven analytics over dynamic workloads with the expected query execution rate in CS being bounded:

**Theorem 3.** *The expected query execution rate in the CS is bounded by* $O(\frac{1}{\lambda^2}(2 - (1-\beta)^{n-1}))$.

**Proof.** Consider that at a certain time instance, there are $n-1$ affiliate ADs which enter in their buffering mode with entry probability $\beta$. Then, the probability of existing at least one affiliate of an AD in the CS in the buffering mode is $1 - (1-\beta)^{n-1}$. Given that a query is forwarded with upper probability $O(\frac{1}{\lambda^2})$ for those ADs being in the buffering mode, then, the expected number of queries being executed in the CS from an AD and its affiliates within any arbitrary time interval $T$ is $T(\frac{1}{\lambda^2})(1+1-(1-\beta)^{n-1})$. □

## 5. Performance evaluation & comparative assessment

The main questions we are striving to answer in our evaluation are the following :

1. How accurate are the given predictions for a variety of aggregate queries?
2. Is there a single ML model that can be used for this purpose?
3. What are the effects of predicates and data dimensionality (number of columns) on estimating the results of aggregate queries?
4. How light-weight and efficient are the models and can they be stored on ADs for efficient execution
5. How effective are the CDM/ADM mechanisms and what is the effect of continuously learning and adapting to new queries?

### 5.1. Implementation & experimental environment

To implement our algorithms we used `scikit-learn` , `XGBoost`[14] and an implementation of the `Growing-Networks` algorithm [15]. We performed our experiments on a desktop machine with a Intel(R) Core(TM) i7-6700 CPU @ 3.40 GHz and 16GB RAM. For the real datasets, the GrowingNetworks algorithm was used for clustering mainly because of its invariance to selecting a pre-defined number of clusters and its ability to naturally grow (as required by our adaptability mechanisms). XGBoost was used as the supervised learning algorithm because of its superior accuracy to other algorithms we tested and shown as part of our evaluation.

Real datasets: We use the **Crimes** dataset from [16] and the **Sensors** dataset from [17]. The Crimes dataset contains $|\mathcal{R}_1| = 6.6 \cdot 10^6$ and the Sensors dataset $|\mathcal{R}_2| = 2.3 \cdot 10^6$ data vectors. These two datasets are widely known in the community of data

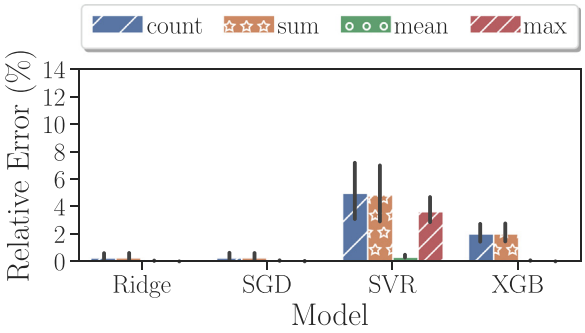



**Fig. 5.** Relative prediction error for descriptive statistics.

management [18–20] and are example datasets that analysts would use during exploratory analysis. Please note that dataset Crimes refers to a set of recorded incidents and a number of associated attributes and Sensors includes recordings over a set of sensors deployed in a lab setting. Analyzing such datasets or other datasets exhibiting the same properties (IoT readings, spatio-temporal etc.) is one of the most common use cases in the case of analysts. In addition, it is expected that the interests of different analysts would vary when exploring these datasets which means that the trained models would need to be adaptive. We also created synthetic query workloads over these datasets as real query workloads do not exist for this purpose as also attested by [4]. For Crimes, we generated predicates restricting the spatial dimension and for Sensors the temporal dimension as essentially this is what analysts would be doing in exploration tasks. For the predicates in the spatial dimension we used multiple multivariate-normal distributions to simulate the existence of multiple users. For the temporal dimension, we used a uniform distribution. We then recorded the results of the descriptive statistics `COUNT`, `MEAN`, `SUM` over different attributes in the datasets to sufficiently make sure the workload is randomized.

**Synthetic datasets**: We also generated synthetic datasets and workloads to stress test our system. We generated a varying number of predicates and attributes to see how it would affect a state-of-the-art model chosen by an initial study comparing different models under different aggregates. This helped us understand the implications of our chosen representation and identified under what conditions the accuracy deteriorates.

### 5.2. Predictability

We measured the prediction accuracy of our system using both synthetic and real datasets. We examine the median relative error, unless stated otherwise. We first examine the *predictability* of different descriptive statistics using a variety of supervised ML models. The experiments were conducted using our synthetic query workloads to test accuracy across a varying number of attributes (columns) and predicates. The models are trained using 80% of the total queries ($=10^4$) and tested against 20%. Where: `Ridge` is for Regularized Linear Regression model, `SGD` is a linear regression model trained on-line using SGD, `SVR` is for Support Vector Regression with RBF kernel, and `XGB` is the XGBoost model. None of the models was hyper-tuned to provide the purest accuracy as we desired to test whether different statistics could be better estimated by different models, thus, indicating the need to choose optimal models at training.

Fig. 5 shows the results of this experiment. The main takeaways are as follows. First, query-driven learning consistently produces low relative errors across all statistics. Second, it is largely insensitive to underlying ML models; given our proposed representation, all models are able to predict the given statistics with small error, well below 10%. Third, there is some high variation across the reported error as all of the workloads with varying predicates and columns were used. Finally, all statistics can be adequately predicted by a single algorithm, that being `XGB`. The latter represents an advantage for this work, as the system can be optimized for storing such models and can be designed around this single class of model instead of trying to accommodate a variety of them, each with its own restrictions.

Using the most accurate model derived from our experiments (`XGB`), we evaluated the prediction accuracy over different statistics under different conditions, shown in Fig. 6. As expected, the increase in number of predicates used and number of columns has a negative impact on accuracy. The representation used is high dimensional as the query vector size is $\mathbb{R}^{2d}$, where $d$ is the number of columns in a dataset. Hence, for $d = 100$ columns used, Fig. 6(right), the model is trained using $\mathbb{R}^{200}$. This is no trivial task and is stress testing our system's capabilities. In addition, the number of predicates used is the number of $(l_i, u_i)$ elements which restrict the sub-space and are sparsely populated. Again this can make the fitting process harder. However, even under these conditions, the increase in error is no more than linear w.r.t to increasing the number of predicates and columns. With statistics such as `MAX`, being less impacted by this change, as their error seems to be invariant from the beginning. Nonetheless, the results are reassuring: The proposed approach delivers very low errors, even when large number of predicates and columns are used in queries. To put things in perspective, the median number of columns selected in a query is around 8 [21] in typical workloads and also followed by our proposed workload.

We also experiment with real datasets to demonstrate the applicability of our system under real conditions. As evident from Fig. 7 our system provides estimations for descriptive statistics over different types of real datasets with relative error below 10%. Note that a relative error below 10% is the target of modern state of the art approximate-answer production systems [21]. We also compared the performance of our system with VerdictDB [22], i.e., a state-of-the-art system in Approximate Query Processing, over the same datasets.

The errors obtained by VerdictDB varied from 1%–14% with sampling ratio of 1%–10%. These results are comparable to ours and show that our system can be reliably used in parallel to such engines when local access is needed and resource consumption at CS is to be minimized. After training the system with more than 2000 queries, the relative error starts approaching its minimum value rather swiftly for both datasets, namely **Crimes** and **Sensors**. This demonstrates the capability of the proposed learning approach to offer high accuracy estimates for approximating analytical query answers with only a fairly small number of training queries. Note that typical industrial-strength in-production big data analytics clusters used for approximating answers to such analytical queries receive several million of queries per day [21]. Therefore, one can expect that a system employing our approach would receive a few thousand of training queries in a just few tens of seconds.

In addition, we also plot the NRSMD error for our proposed system and VerdictDB [22]. The NRMSD error is computed as follows :

$$NRMSD = \frac{RMSE}{\bar{y}} \tag{6}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y - \hat{y})^2} \tag{7}$$

NRMSD is a ratio interpreted as an error in which the lower the better. We investigate the NRMSD for the aggregates `COUNT`,

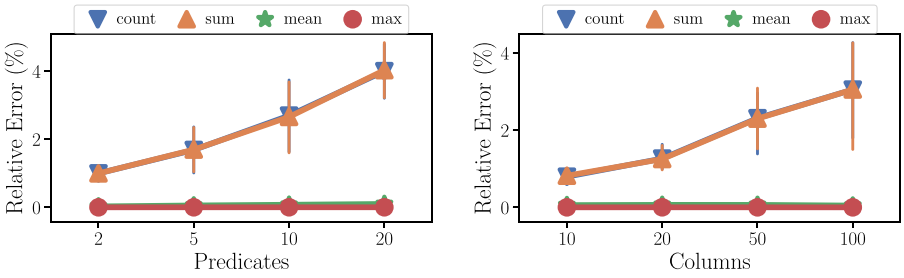



**Fig. 6.** Relative error of statistics vs. increasing number of (left) predicates, (right) columns.

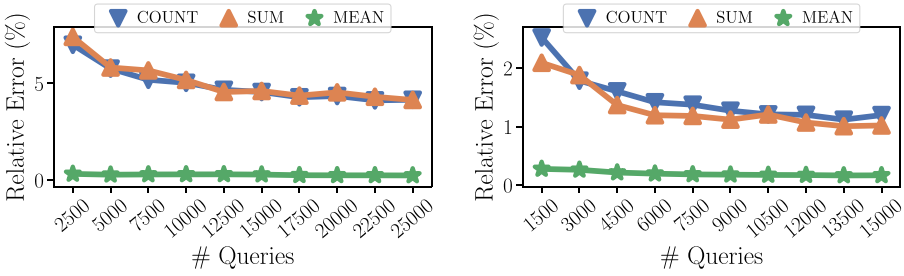



**Fig. 7.** Relative error vs. number of training queries; (left) Crimes (right) Sensors.

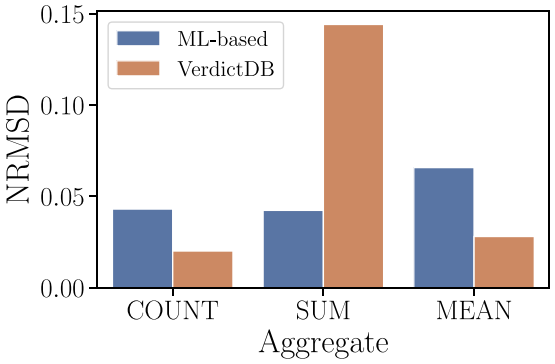



**Fig. 8.** Accuracy comparison of our approach against a state-of-the-art sampling based AQP system, VerdictDB [22].

MEAN/AVG, SUM under the Crimes dataset using both our approach and VerdictDB. As is evident at Fig. 8 the errors are similar and do not exceed 15%. Finally, the above experiments show that our approach is accurate in predicting the results of different aggregate functions and that is on par with state-of-the-art sampling based AQP engines that are designed specifically for this purpose.

### 5.3. Performance & storage

We examine the performance and storage requirements of our system. This is important as our solution has to be light-weight both in terms of storage overhead for ADs and efficient in transferring models through the network. We examine all the above-mentioned models to identify the most efficient ones in training and prediction. A synthetic workload with 50 predicates and 100 columns is used for training all the models. For *Prediction Time* (PT) in Table 1 we report on the expected prediction time and standard deviation of each model. As expected, SVR has the worst performance. The central takeway here is that PTs are negligible — much less than a millisecond, thus, guaranteeing efficient statistics estimation irrespective of the adopted model. Even though there are multiple models trained, to account for varying query patterns, the time complexity associated is $\mathcal{O}(K)$ with $K$ usually being small. We show that our learning models based on mining query logs and learning using query-answer pairs, can go a long way in amplifying the capabilities of analytic system stacks as they can act as a 'caching' mechanism without actually storing the results of past queries but instead using models to perform answer estimation for new queries.

**Table 1**
Performance and storage results across models.

| | Size (kB) | TT (s) | PT (ms) |
|---|---|---|---|
| Ridge | $1.45 \pm 0.6$ | $0.55 \pm 1$ | $0.0008 \pm 0.0004$ |
| SGD | $1.73 \pm 0.6$ | $0.44 \pm 0.8$ | $0.0008 \pm 0.0007$ |
| SVR | $1332.68 \pm 944$ | – | $0.14 \pm 0.078280$ |
| XGB | $65.42 \pm 4$ | $52.58 \pm 90$ | $0.008 \pm 0.005201$ |

For measuring the training time of individual models, we varied the number of training samples and examined the expected model Training Time (TT) in Table 1. We used 11 training samples varying in size w.r.t. $\{4 \cdot 10^2, \ldots, 4 \cdot 10^5\}$. For SVR, we stop recording after $1.6 \cdot 10^4$ training samples as the algorithm is no longer efficient and should be avoided. Although XGB appears to perform the worst, we note that its TT is no more than 53 seconds without using its multi-threading capabilities.

We also, examine the model Size in kB shown in Table 1 and observe that it refers to a negligent cost fulfilling our initial requirements about the system being light-weight to reside in the ADs memory. The results are for one individual model therefore the resulting storage cost is $K$ times the initial one. This could be a significant overhead, especially for SVR as $K \gg 1000$. However, the cost incurred is not preventive as the benefit of decreasing latency times, offloading queries otherwise issued to the cluster and no extra monetary cost are far greater.

### 5.4. Adaptivity

To examine our CDM, ADM due to concept drift we devised the following experiment: consider a $\mathcal{M}_i$ that has already learned a particular distribution of $y$ being deployed to answer queries. At a particular point in time the query patterns might shift as shown in Fig. 9. Fig. 9 shows two different query distributions. Fig. 9(left) are the distributions of the query answers $y$. Their

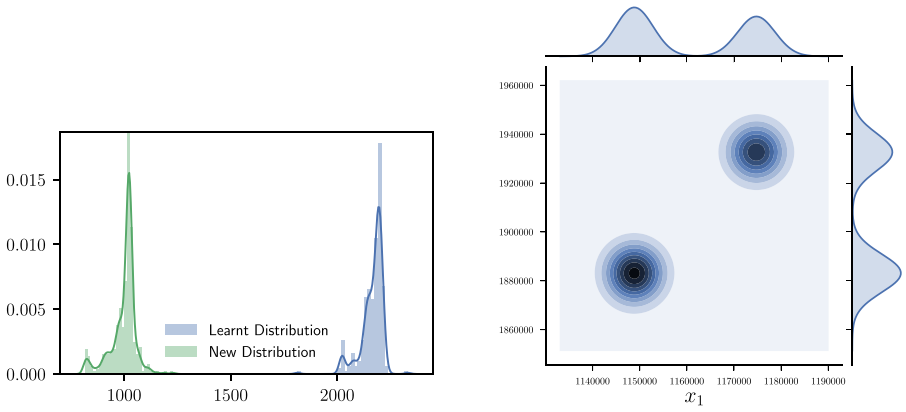



**Fig. 9.** (Left) Inducing concept drift of learned distribution of $y$; (right) Lower left is the initial query pattern distribution; upper right is the new one.

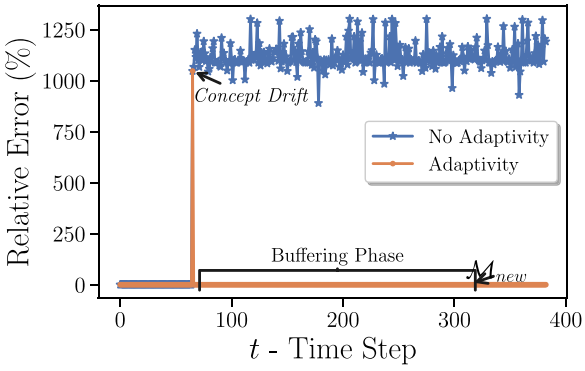



**Fig. 10.** Error with concept drift detection/adaptation.

respective query patterns are shown in Fig. 9 (right).[4] Our aim is to examine whether CDM detects a query pattern shift from one distribution to another. If remained undetected, it will cause detrimental problems in accuracy due to different distributions of $y$. We first set the detection threshold $h = 3\sigma_{\tilde{u}}$ and convergence threshold $c = 0.008$; a following sensitivity analysis shows the impact of these tuning parameters on ADM and CDM. We gradually introduce new query patterns and compare our system with an approach where no adaptation is deployed. Fig. 10 shows queries being processed by our mechanism and the associated *true* prediction error. We first measure the error of queries using the *known* distribution until $t = 66$. From that point onwards, we shift to the unknown distribution and evidently the error increases dramatically should no adaptation mechanism be employed. On the other hand, CDM detects that a shift has happened and transits the system from prediction mode to buffering mode until the exiting criteria are met. At the end, a new model is introduced which is trained using the new distribution as evidenced by the decreased error at $\mathcal{M}_{new}$.

Parameters $h$ and $c$ are responsible for the ADM and CDM with the impact of $c$ shown in Fig. 11(left). As we increase $c$ we allow for an early exit from buffering mode. An early exit means that less queries have been processed thus potentially the examples are not sufficient for accurately learning the distribution as witnessed in Fig. 11(left), where the relative error increases, therefore the accuracy decreases as we increase $c$.

Fig. 11(right) shows the diminishing probability of entering the buffering mode $\beta = P(G > h)$ building upon our discussion of slowly converging the system into an Offline mode. As more queries are processed across varying query spaces then our system is incrementally learning the whole data space. At a certain point, all query subspaces will be *known* along with their representatives. Thus, the probability of entering the buffering mode due to potentially unknown query distribution reduces to zero almost surely. We provide an experiment in which there is a predefined fixed number of Query Spaces (QS) $K = 16$, $QS = \{QS_1, \ldots, QS_K\}$. Queries are generated randomly in a sequence from one QS to another, each time learning $QS_{k-1}$. Thus, given this fixed number of QSs and set $h, c$, the probability of entering buffering mode can be approximated by $\beta = P(G > h) = 1 - \frac{l}{K}$, where $l \leq K$ denotes the number of known QS so far. Liaising this with Fig. 11, we observe that the probability reduces in a step-wise manner tending to zero when $l \to K$. For a relatively high $c$ value, the rate of convergence to the offline mode becomes faster but with a higher error as witnessed by the previous experiment. As for parameter $h$, a low value indicates smaller tolerance when estimating errors and vice versa. This might force the system to adapt when not needed. Thus, it is domain appropriate to hyper-tune the parameter accordingly; hyper-tuning $h$ is part of our on-going research. However, we have found the proposed heuristic of $3\sigma \leq h \leq 5\sigma$ to work well empirically.

### 5.5. CDM sensitivity analysis & robustness

We also conducted experiments to analyze the sensitivity of the CDM to various properties focusing on (a) noisy observations, (b) outliers that might be encountered as part of the original distribution and do not indicate a distribution shift, and (c) skewed distributions in the predictor variables. By examining these properties, one can be advised on the robustness of the CDM.

We initially conducted an experiment to quantify the impact of the noise on the CDM performance. Specifically, we repeat the same experiment as described in Section 5.4. Queries, and hence their estimated errors are sampled from an initial distribution $p_1$, where $p_1(\tilde{u})$ is the distribution of the estimated errors that are calculated using (2). We then introduce white noise into these estimations such as $\tilde{u} \leftarrow \tilde{u} + \mathcal{N}(0, v^2)$, where $N(0, v^2)$ is Gaussian distribution with zero mean value and finite variance $v^2$ for $v \in [0, 3]$. We repeat this experiment for 200 random time-points $t^*$, where a time-point $t^*$ is the time instance at which we switch distribution $p_1(\tilde{u})$ to $p_2(\tilde{u})$, which is expected to trigger the CDM. We then record the exact time-point $t_D$ where the CDM has actually fired and calculate their expected detection difference of $t^* - t_D$, notated as $\mathbb{E}[t^* - t_D]$. Notice that we do not calculate their absolute value difference, as a negative value indicates that the CDM has fired after $t^*$, while a positive value before $t^*$. As before,

[4] Only two dimensions shown for visualization purposes.

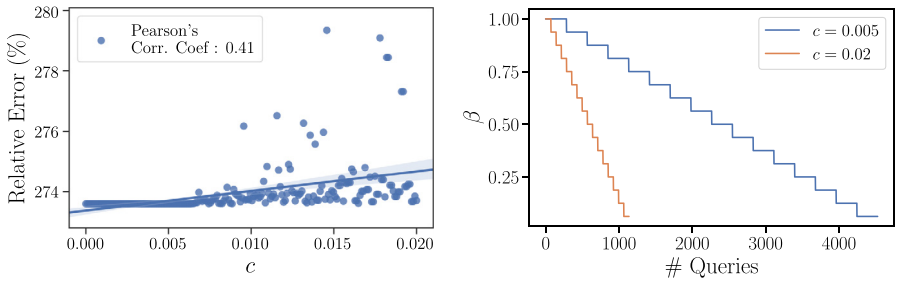



**Fig. 11.** (Left) Error vs. convergence $c$; (right) diminishing probability of buffering $\beta$ as more query spaces are *known*.

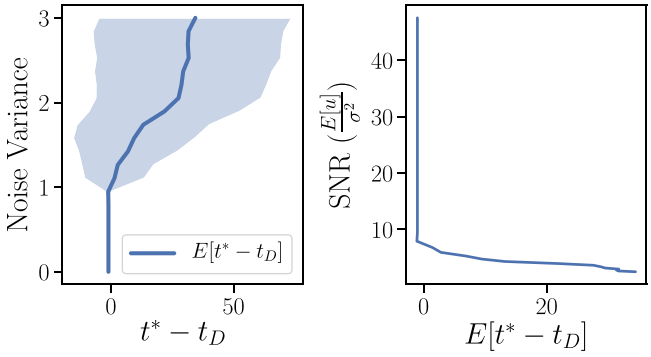



**Fig. 12.** (Left) Expected time-point detection difference (x-axis) as the noise intensifies (y-axis). (Right) The expected time-point detection difference with respect to the Signal-to-Noise-Ratio.

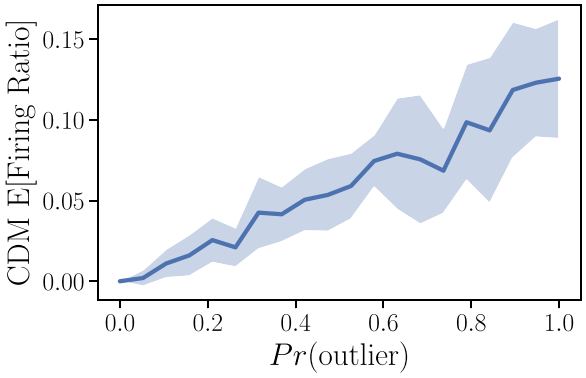



**Fig. 13.** The $x$-axis shows the probability of injecting an outlier to the estimated errors, while the $y$-axis shows the expected firing ratio for the CDM (lower is better).

we set the threshold for $h = 3\sigma$, where $\sigma$ is the standard deviation of $p_1(\tilde{u})$. The results for this experiment are shown at Fig. 12. Fig. 12(left) shows the expected time-point difference, $\mathbb{E}[t^* - t_D]$ and the standard deviation across many runs. It is evident that as the noise variance is within $[0, 1]$ the expected time-point detection difference is 0 which indicates that the CDM is robust to noise. However, as the noise intensifies, it effectively shifts the distribution for the estimations $\tilde{u}$ and the CDM fires prematurely on average. This is due to the fact that the noisy observations are now deemed as large error estimations. Nonetheless, this is not to say that the CDM is no longer robust, as we expect this kind of behavior. This is reinforced by measuring the Signal-to-Noise-Ratio (SNR), which we plot in Fig. 12(right). The SNR is a quantity that measures how much stronger the signal is with respect to the noise. As it is evidenced in Fig. 12(right), the expected time-point detection difference only increases as the SNR decreases, which indicates that the noise becomes really large and then weakens the signal.

We have also investigated the effects of outliers on CDM. We sample values from an initial distribution $p_1(\tilde{u})$ and record whether the CDM has fired. However, we introduce outliers to the sampled values by modifying the estimated error using $\tilde{u} = \tilde{u} + m\sigma$, where $\sigma$ is the standard deviation of the distribution for $\tilde{u}$ and multiplicative factor $m \in \{3, 4\}$. The value of $m$ is selected based on a Bernoulli trial with success probability 0.2 and a success sets the value of $m = 4$. An outlier is *injected* based on a Binomial distribution with probability of success $p$ over a number of trials $n = 100$, where $p \in [0, 1]$. We vary the probability $p$ and record the expected firing ratio for a number of runs. We plot the results of this experiment in Fig. 13. As one can observe, the expected firing ratio is relatively low even for outliers probability $Pr\{\text{outlier}\} = 0.5$, while the expected firing ratio is less than 15% where the $Pr\{\text{outlier}\} = 1$. We can also observe a linear increasing trend on the expected firing ratio with respect to the probability of injecting an outlier. Please note, that even at $Pr\{\text{outlier}\} = 1$, the CDM should not fire as the outlier values for $\tilde{u}$ do not indicate a distribution shift, but *extreme* values of the same distribution. Hence, given this experiment it is safe to conclude on the fact that CDM is robust to outliers.

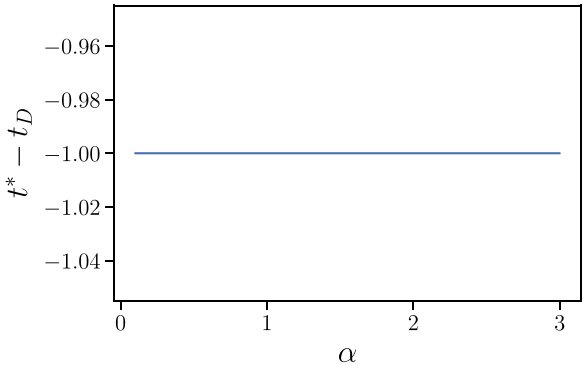



**Fig. 14.** The parameter $\alpha$ controls the skewness of the $\Gamma(\alpha, \beta)$ distribution; the skew $= \frac{2}{\sqrt{\alpha}}$ has no visible effect the on CDM.

Lastly, we evaluate the CDM robustness and performance for various highly skewed distributions of the predictors. In this experiment, our aim is to examine whether skewed distributions for the predictors would affect the CDM's accuracy. As the CDM largely relies on a distance-based metric (refer to (2)), i.e., CDM is highly influenced by the predictors, a skewed distribution might affect the estimations produced and might cause the CDM to misfire. For this experiment, we use $\Gamma(\alpha, \beta)$ distributions for the predictors. Each predictor is distributed with respect to $\mathbf{q} \sim \Gamma(\alpha, \frac{\beta}{c})$, where $\alpha \in [0.1, 3]$ and parameters $\beta$ and $c$ are pre-defined values that define the *scale* and *location* of the distribution. Therefore, we initially generate a distribution $p_1(\mathbf{q})$ where $\mathbf{q} \sim \Gamma(\alpha, \frac{\beta_1}{c_1})$. We set the EPE $(u_k)$ to a constant and

obtain an initial distribution of estimated errors $p_1(\tilde{u})$ based on (2). We also generate a second distribution of predictors $p_2(\mathbf{q})$ where $\mathbf{q} \sim \Gamma(\alpha, \frac{\beta_2}{c_2})$ and obtain the distribution of errors $p_2(\tilde{u})$ that we consider as the errors when the distribution of queries has shifted. We then perform the same experiment as before, initially the error estimates from $p_1$ are passed to the CDM. Then at a random point in time, we switch to $p_2$ and observe the detection time-point difference $t^* - t_D$. We plot the results of this experiment in Fig. 14. The $\alpha$ parameter controls the skewness of the predictors distributions. The skewness decreases at a rate of $\frac{2}{\sqrt{a}}$. As it is evidenced, the CDM is unaffected by this as it correctly fires immediately after the time-point $t^*$ where the shift happens.

Overall, through these set of experiments, we have examined the CDM's sensitivity on a number of different settings. We have demonstrated its applicability and robustness under various conditions and concluded on the fact that CDM is appropriate to use in many scenarios.

## 6. Related work

Our work is related to prior work in applied ML research communities and to prior work focusing on the benefits of the query-driven approach in analytical query processing and tuning [18,23–26]. Specifically, the authors in [23,24] apply statistical learning models (fusing unsupervised with supervised ML schemes) only for selectivity estimation, while the model in [25] deals with approximate query processing. The method proposed in [26] focuses on database tuning, and the ML model developed in [18] is used to provide explanations on the outcome of aggregate queries. The above works follow a similar methodology in applying ML over data management problems. Our work follows a similar approach with *one* fundamental difference; we explicitly address the case where the input distribution of the models shifts, i.e., the underlying distribution is subject to changes. This is a principal issue to all works applying ML algorithms over data management problems. Hence, our proposed adaptation mechanisms can be utilized by any of these existing works to alleviate the issue of changing input distributions as efficiently as possible.

Analytical queries nowadays are executed over underlying systems that provide either exact answers [27,28] or approximate answers [21,22,29,30] working over large big data clusters in DCs/CS requiring several orders of magnitude longer query response times. Specifically, the work in [22,29,30] is based on accessing and sampling large databases offline and subsequently providing approximate answers to analytical queries. This is a common way of attempting to provide interactive analytics capabilities to the users. However, such solutions still have a large storage overhead as the size of samples is directly linked to the size of datasets. On the other hand, the authors in [21] introduce a data-access model to inject samples in the query plan at runtime. This does not require storing the samples or incur the initial cost of sampling a database. Nevertheless, it significantly increases the latency at which the answers are served back to the users/analysts. The contributions in this work are largely complementary to all these works. Specifically, during the training phase and in the adaptation/buffering phase, the system proposed here can be supported either by an exact or an approximate query processing engine. In addition, what makes our solution different is that it can be stored locally on analysts' devices as it has low storage overhead and also requires no communication to the cluster. Moreover, it requires no access to data and relies on previous queries to train ML models, which are then used to answer analytical queries.

Query-driven models are largely being deployed for both aggregate estimation [23,24] and for hyper-tuning [31] database systems. Unlike the models in [23,24], our focus is on a wide variety of aggregate operators and not just the COUNT operator for selectivity estimation. Furthermore, we address the crucial problem of detecting query pattern changes and adapting to them, which (to our knowledge) has not been addressed in this context before. Hence, our framework can be leveraged by all query-driven implementations in cases of dynamic workloads that are non-stationary.

Moreover, concept drift adaptation is well understood [9–11,32], mostly dealing with classification tasks, where classifiers adapt to new classes. The authors of [9] introduce the problem of concept drift and elaborate on subtle differences of the different forms of concept drift. Specifically, different methods have to be applied when the input or output distributions of an ML model change. Although our solution focuses on changes in the input distribution we only have to adapt in cases where the output distribution is affected, as explained in previous sections. In addition, works such as [10,11,32,33], describe on a number of approaches that can be utilized for adapting under the presence of concept drift for a number of different scenarios. However, none of the works focus on data management problems or adapting the algorithms to work with approximate signals indicating concept drift. We adapt concept drift to query-driven analytical processing, relying on *explicit partitioning* [11], ensuring it avoids destructive forgetting given that the accuracy for the previously learned query patterns will not degrade. It is also favorable given our initial off-line design which already uses partitioning for clustering the query patterns and learning local models in given sub-spaces. Our work contributes to monitoring and detecting real-time query patterns change based on *approximating* the prediction error, which differentiates with the previous concept drift methods by measuring the actual error. Evidently, this is not applicable in our case. Finally, we propose a novel reciprocity-driven adaptation mechanism in which we set a mechanism deciding when a new model should be trained engaging the knowledge derived from other possibly changing models in the CS.

## 7. Conclusions & future work

In this work we contribute a novel framework for adapting trained models under concept drift of the underlying query workload distribution. We focus on lightweight on-line statistical learning models used for estimating analytical query answers efficiently and accurately, however we note that the framework is applicable in other domains as well.

The contributions center on a novel suite of ML models, which mine past and new queries and incrementally build models over quantized query-spaces using a vectorial representation. The introduced mechanisms (ADM and CDM) bear the ability to adapt under changing analytical workloads, while maintaining high accuracy of estimations. As shown by our evaluation (using real and synthetic datasets), the proposed approach enjoys high accuracy (well below 10% relative error) across all aggregate operators, with low response times (well below a millisecond) and low footprint and training-time overheads. The contributed adaptability mechanism is able to detect changes using estimated errors and swiftly adapts. Furthermore, as more queries are processed, our system has the potential to reach global convergence as no more query patterns remain undiscovered. This can significantly reduce unnecessary communication to cloud providers thus reduce network load and monetary costs.

As part of our future work we are looking to apply the principles of this methodology for identifying *when* and *how* to adapt models in a data management setting under various contexts. This includes situations where ML algorithms are applied, for example, to explaining aggregates [18], handling automatic database tuning [31], and learning multi-dimensional indexes

[34]. In addition, it will be of high importance to explore if and when complete knowledge of all the transitions in query-space actually happens. Specifically, as data analysts continue to explore the initially unknown data-space, more and more queries are executed in different data subspaces causing the models to shift, adapt, and thus learn new data subspaces of potential interest. It could be the case that at a certain point in time there is nothing more to explore, thus, the system can then fully rely on the trained models. It will be paramount to be able to identify when this will happen.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## CRediT authorship contribution statement

**Fotis Savva:** Conceptualization, Methodology, Software, Validation, Investigation, Writing - original draft, Writing - review & editing, Visualization. **Christos Anagnostopoulos:** Conceptualization, Methodology, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition. **Peter Triantafillou:** Conceptualization, Writing - original draft, Writing - review & editing, Supervision.

## Acknowledgments

This work is partially funded by the European Union EU/H2020 Marie Sklodowska-Curie Action Individual Fellowship (MSCA-IF) under the project INNOVATE (Grant No. 745829) and the UK EPSRC grant EP/R018634/1.

## Appendix

### A.1. Real datasets & workloads

**Crimes:** For constructing the workload over the **Crimes** dataset we initially sampled 10k points and obtained the mean and standard deviation for the attributes *X_Coordinate*, *Y_Coordinate*. By setting the obtained statistics as parameters for a multivariate normal distribution we generated a number of *Cluster* points (=5). Centered on each one of those cluster points we constructed multivariate normal distributions with a fraction (=0.01) of the original standard deviation. Based on this we generated a number of points (=10k) for around each cluster point. Based on those points we constructed queries covering a random range across both the *X/Y Coordinates*. For each point a fraction of the original std was used to construct ranges : x_range $= \frac{\text{x_std}}{2}$rand(0,1), where *x_std* is the original standard deviation, same approach was followed for the *y_range*. The queries filtered the complete dataset based on (where pt is the point):

$$\alpha_{X\_Coordinate} \geq \text{pt} - \text{x\_range} \wedge \alpha_{X\_Coordinate} \leq \text{pt} + \text{x\_range}$$
$$\wedge \alpha_{Y\_Coordinate} \geq \text{pt} - \text{y\_range} \wedge \alpha_{y\_Coordinate} \leq \text{pt} + \text{y\_range} \quad (8)$$

On the filtered dataset generated by each query with varying cardinality we extracted basic statistics like `COUNT`, `AVG`, `SUM` on attributes that would make sense (*Beat* - avg, *Arrest* - sum).

**Sensors**: For Sensors we obtained the mean and standard deviation of the temporal dimension after encoding it and normalizing it. The min/max statistics were also obtained. We then generated a number of queries (=50k) with range equal to a fraction of the complete distance between the max/min ($0.2 \times$ (max $-$ min). The center of the queries was randomly generated by a normal distribution with parameters equal to the obtained mean and half original standard deviation. The same processes as in Crimes was used to filter the complete dataset and extract statistics on attributes that made sense (*Temperature*-avg, *Light*-sum).

### A.2. Synthetic data generation

Specifically, we generated datasets with varying dimensionality $[10, 20, 50, 100]$ to simulate data with large number of columns. Each column contains numbers generated from a uniform distribution $\mathcal{U}[0, 10^6]$, therefore each point is $\mathbf{x} \in [0, 10^6]^d$ where $d \in [10, 20, 50, 100]$. We then generated a number of query workloads with a varying number of selected columns to be restricted by predicates $p \in [2, 5, 10]$ (Increasing predicates even further resulted in no tuples returned) according to the dimensionality of each of the synthetic datasets. For instance, for a dataset with $d = 10$ the resulting workloads were two, with the selected columns being (2, 10). Each query point for those two datasets is then $\mathbf{q} \in [0, 10^6]^{2p}$.

**Algorithm 2:** Generating Queries

**Input** : $N$,# queries, $\mathcal{D} = [10, 20, 50, 100]$, # columns
**Input** : $\mathcal{P} = [2, 5, 10]$, # predicates (selected columns)
**Input** : $r$, range size retrieved from histogram to ensure similar selectivities

**for** $d \in \mathcal{D}$ **do**
  **for** $p \in \mathcal{P}$ **do**
    **for** $i = 1$ ***to*** $N$ **do**
      $\mathbf{z} \leftarrow random(\mathcal{N}(10^6, 100), p)$ ;
      $\mathbf{b} \leftarrow randomBit([0, d], p)$ ;
      $\mathbf{lb} \leftarrow \mathbf{z} - \frac{r}{2}$ ;
      $\mathbf{ub} \leftarrow \mathbf{z} + \frac{r}{2}$ ;
      $\mathbf{q}_i^{(m)} \leftarrow [\mathbf{lb}, \mathbf{ub}]$;
      $y_i \leftarrow executeAggregates(\mathbf{q}_i^{(m)}, d)$ ;
      $\mathbf{q}_i \leftarrow (\mathbf{q}_i^{(m)}, y)$ ;
      $\mathcal{W}_{d,p} \cup \{\mathbf{q}_i\}$;
    **end**
  **end**
**end**

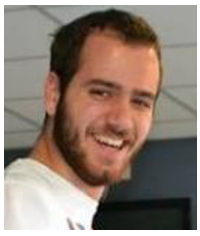
**Output**: Resulting workloads
$\mathbf{W} = (\mathcal{W}_{10,2}, \ldots, \mathcal{W}_{d,p}), \forall d \in \mathcal{D}, \forall p \in \mathcal{P}$

We also describe the query generation process in Algorithm 2. We first set the number of queries $N$ and number of predicates and columns in our datasets $\mathcal{D}$, $\mathcal{P}$. The retrieved range size $r$ is obtained by a Normal distribution $\mathcal{N}(\text{sel} \cdot 10^6, 0.01 \cdot 10^6)$ with the mean being a fraction of the complete range defined by selectivity ratio $sel = 0.5$ to ensure enough tuples are returned as with an increasing number of predicates less tuples are returned. We then loop through over the parameters issued and generated queries using the provided algorithms. Where the *random* function returns a random vector of size $p$ over the given range, and *randomBit* generates a random bitmask for the selection of columns taking part in the query. Then the lower-bound and upper-bound for the given predicates is set and we execute the query over the restricted space given by the bounds over the selected columns by $b$ using dataset $d$.

## References

[1] Z. Liu, J. Heer, The effects of interactive latency on exploratory visual analysis, IEEE Trans. Vis. Comput. Graph. (1) 1.
[2] S. Idreos, O. Papaemmanouil, S. Chaudhuri, Overview of data exploration techniques, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 277–281.

[3] A. Vulimiri, C. Curino, P.B. Godfrey, T. Jungblut, K. Karanasos, J. Padhye, G. Varghese, Wanalytics: Geo-distributed analytics for a data intensive world, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, 2015, pp. 1087–1092.
[4] A. Wasay, X. Wei, N. Dayan, S. Idreos, Data canopy: Accelerating exploratory statistical analysis, in: Proceedings of the 2017 ACM International Conference on Management of Data, ACM, 2017, pp. 557–572.
[5] S. Masoudnia, R. Ebrahimpour, Mixture of experts: a literature survey, Artif. Intell. Rev. 42 (2) (2014) 275–293.
[6] J. Friedman, T. Hastie, R. Tibshirani, The Elements of Statistical Learning, Vol. 1, Springer series in statistics, New York, NY, USA, 2001.
[7] Growing networks, 2018, URL: https://github.com/Skeftical/GrowingNetworks, Accessed: 2018-08-10.
[8] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, D. Dennison, Hidden technical debt in machine learning systems, in: Advances in Neural Information Processing Systems, 2015, pp. 2503–2511.
[9] A. Tsymbal, The problem of concept drift: definitions and related work, Comput. Sci. Dep. Trinity Coll. Dublin 106 (2) (2004).
[10] G. Ditzler, M. Roveri, C. Alippi, R. Polikar, Learning in nonstationary environments: A survey, IEEE Comput. Intell. Mag. 10 (4) (2015) 12–25.
[11] A. Gepperth, B. Hammer, Incremental learning algorithms and applications, in: European Symposium on Artificial Neural Networks (ESANN), 2016.
[12] O.A. Grigg, V. Farewell, D. Spiegelhalter, Use of risk-adjusted cusum and rsprtcharts for monitoring in medical contexts, Stat. Methods Med. Res. 12 (2) (2003) 147–170.
[13] L. Bottou, Stochastic gradient descent tricks, in: Neural Networks: Tricks of the Trade, Springer, 2012, pp. 421–436.
[14] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 785–794.
[15] S. Marsland, J. Shapiro, U. Nehmzow, A self-organising network that grows when required, Neural Netw. 15 (8–9) (2002) 1041–1058.
[16] Crimes - 2001 to present, 2018, URL: https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2, Accessed: 2018-08-10.
[17] Intel lab data, 2018, URL: http://db.csail.mit.edu/labdata/labdata.html, Accessed: 2018-08-10.
[18] F. Savva, C. Anagnostopoulos, P. Triantafillou, Explaining aggregates for exploratory analytics, in: 2018 IEEE International Conference on Big Data (Big Data), IEEE, 2018, pp. 478–487.
[19] F. Savva, Query-driven learning for next generation predictive modeling & analytics, in: Proceedings of the 2019 International Conference on Management of Data, 2019, pp. 1844–1846.
[20] E. Wu, S. Madden, Scorpion: Explaining away outliers in aggregate queries, 2013.
[21] S. Kandula, A. Shanbhag, A. Vitorovic, M. Olma, R. Grandl, S. Chaudhuri, B. Ding, Quickr: Lazily approximating complex adhoc queries in bigdata clusters, in: Proceedings of the 2016 International Conference on Management of Data, ACM, 2016, pp. 631–646.
[22] Y. Park, B. Mozafari, J. Sorenson, J. Wang, Verdictdb: universalizing approximate query processing, in: Proceedings of the 2018 International Conference on Management of Data, ACM, 2018, pp. 1461–1476.
[23] C. Anagnostopoulos, P. Triantafillou, Learning set cardinality in distance nearest neighbours, in: Data Mining (ICDM), 2015 IEEE International Conference on, IEEE, 2015, pp. 691–696.
[24] C. Anagnostopoulos, P. Triantafillou, Query-driven learning for predictive analytics of data subspace cardinality, ACM Trans. Knowl. Discov. Data (TKDD) 11 (4) (2017) 47.
[25] L. Ma, D. Van Aken, A. Hefny, G. Mezerhane, A. Pavlo, G.J. Gordon, Query-based workload forecasting for self-driving database management systems, in: Proceedings of the 2018 International Conference on Management of Data, ACM, 2018, pp. 631–645.
[26] K.-U. Sattler, I. Geist, E. Schallehn, Quiet: Continuous query-driven index tuning, in: Proceedings of the 29th International Conference on Very Large Data Bases-Volume 29, VLDB Endowment, 2003, pp. 1129–1132.
[27] S. Melnik, A. Gubarev, J.J. Long, G. Romer, S. Shivakumar, M. Tolton, T. Vassilakis, Dremel: interactive analysis of web-scale datasets, Proc. VLDB Endow. 3 (1–2) (2010) 330–339.
[28] A. Thusoo, J.S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, R. Murthy, Hive: a warehousing solution over a map-reduce framework, Proc. VLDB Endow. 2 (2) (2009) 1626–1629.
[29] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, I. Stoica, Blinkdb: queries with bounded errors and bounded response times on very large data, in: Proceedings of the 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 29–42.
[30] Y. Park, A.S. Tajik, M. Cafarella, B. Mozafari, Database learning: Toward a database that becomes smarter every time, in: Proceedings of the 2017 ACM International Conference on Management of Data, ACM, 2017, pp. 587–602.
[31] D. Van Aken, A. Pavlo, G.J. Gordon, B. Zhang, Automatic database management system tuning through large-scale machine learning, in: Proceedings of the 2017 ACM International Conference on Management of Data, ACM, 2017, pp. 1009–1024.
[32] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[33] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. (CSUR) 46 (4) (2014) 1–37.
[34] T. Kraska, A. Beutel, E.H. Chi, J. Dean, N. Polyzotis, The case for learned index structures, in: Proceedings of the 2018 International Conference on Management of Data, ACM, 2018, pp. 489–504.



**Fotis Savva** is a Ph.D. student and member of the Essence: Pervasive & Distributed Intelligence Lab, Information Data and Analysis (IDA) Section at the School of Computing Science, University of Glasgow. His expertise lies in the domains of large-scale data analytics systems and how machine learning can be applied to expedite complex query processing and data analytic tasks. Throughout his Ph.D. he has investigated and published work into (a) how Machine Learning models can explain data subspaces to guide exploratory analysis, (b) how Query-Driven learning can be applied for Approximate Query Processing with error guarantees (c) how can the process of identifying interesting subspaces can be expedited by the use of machine learning models that serve as surrogate analytic functions (d) how trained Machine Learning models can be adapted when faced with non-stationary distributions using approximate detection mechanisms. His work on "Query-Driven Learning for Next Generation Predictive Modeling and Analytics" received 2nd place at the Student Research Competition at ACM SIGMOD 2019. In 2016 he was awarded an EPSRC scholarship for his Ph.D. studies. He holds a BSc (Hons) First Class in Computer Science, University of Glasgow.

**Dr Christos (Chris) Anagnostopoulos** is an Assistant Professor of Distributed and Pervasive Computing, School of Computing Science, University of Glasgow. His expertise is in the areas of network-centric pervasive & adaptive systems and in-network information processing in large-scale distributed sensor/UxV/Edge networks. He has received funding for his research by the EU, UK EPSRC, and the industry. Dr Anagnostopoulos is coordinating the projects: EU GNFUV and EU Marie Sklodowska-Curie (MSCA)/INNOVATE, and is a co-PI of the EU PRIMES and UK EPSRC CLDS. Dr Anagnostopoulos is an author of over 140 refereed scientific journals and conferences. His papers have recieved several awards including IEEE Big Data 2018, IEEE WD 2019. He is leading the Essence: Pervasive & Distributed Intelligence Lab within the Information, Data, and Analysis (IDA), University of Glasgow. He serves as an Editorial Member in the Applied Intelligence, Distributed Sensor Networks, and Open Computer Science journals. Dr Anagnostopoulos before joining Glasgow was an Assistant Professor at Ionian University and Adjunct Assistant Professor at the University of Athens and University of Thessaly. He has held postdoctoral positions at University of Glasgow and University of Athens in the areas of in-network context-aware distributed & mobile computing systems. He holds a BSc, MSc, and Ph.D. in Computing Science, University of Athens. He is a member of ACM and IEEE.

**Peter Triantafillou** is Professor of Data Systems and Head of the Data Sciences Theme at the Department of Computer Science at the University of Warwick, Fellow of the Alan Turing Institute , member of the Advisory Board of the Urban Big Data Research Centre (a national infrastructure for urban data services and analytics), and Honorary Senior Research Fellow at the School of Computing Science at the University of Glasgow. Prior to that, Peter held professorial positions at the University of Glasgow, Simon Fraser University in Canada, Technical University of Crete and University of Patras in Greece, and visiting professorships at the Max-Planck Institute for Informatics in Germany. Peter received his Ph.D. in computer science from the University of Waterloo and was the Department of Computer Science and the Faculty of Mathematics nominee for the Gold Medal for outstanding achievements at the Doctoral level.

Peter has published extensively in top journals and conferences in his areas. Peter's papers have received numerous awards, including the most influential paper award in ACM DEBS 2019 Conference for his 2004 DEBS paper, the best student paper award at IEEE Big Data 2018, the best paper award at

the ACM SIGIR Conference (on Information Retrieval) in 2016 and the best paper award at the ACM CIKM Conference (on Information and Knowledge Management) in 2006. Peter has served in the Technical Program Committees of more than 120 international conferences and has been the PC Chair or Vice-chair in several prestigious conferences. Finally, Peter is a co-designer of several innovative systems (such as the MINERVA decentralized search engine and the eXO decentralized social networking system). Currently, our group is developing the DBEst system for large-scale analytics, whose key salient feature is data-less (big) data analytics: where analytical queries are processed through the access of Machine Learning models and without accessing any data.